IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Matthew Marcus		Art Unit	: 2166
Serial No. : 10/716,840		Examiner	: Usmaan Saeed
Filed : November 18, 2003		Conf. No.	: 7089
Title : OPTIMIZATIONS OF XPATHS			

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

The Notice of Appeal in this matter was filed with the United States Patent and Trademark Office through the Electronic Filing System (EFS) on May 25, 2007.

(1) Real Party in Interest

Adobe Systems Incorporated, the assignee of this application, is the real party in interest.

(2) Related Appeals and Interferences

None.

(3) Status of Claims

Claims 1-34 and 37-40 are pending in this application. Claims 1-34 and 37-40 stand rejected. The rejection of claims 1-34 and 37-40 is appealed.

(4) Status of Amendments

There are no un-entered amendments.

(5) Summary of Claimed Subject Matter

Claim 1

Claim 1 recites a method directed towards searching for one or more logical elements in a hierarchical tree structure of an extensible markup language (XML) document conforming to a schema used for XML (page 4, lines 15-21; FIG. 1). A representation of an XML document instance is provided (page 4, lines 15-16). The XML document instance contains two or more logical elements, where at least one logical element is a parent node and at least one logical element is a child node in a hierarchical tree structure describing the representation (page 4, lines 19-21). A query for logical elements satisfying an XPath expression is received (page 6, lines10-11). Only nodes that potentially have child nodes satisfying the XPath expression are searched in the hierarchical tree structure (page 7, lines 1-3 and 18-26). The logical elements that satisfy the XPath expression are provided (page 7, lines 24-26).

Claim 20

Claim 20 recites a computer program product directed towards searching for one or more logical elements in a hierarchical tree structure of an extensible markup language (XML) document conforming to a schema used for XML (page 4, lines 15-21; page 8, lines 3-8; FIG. 1). The computer program product is tangibly encoded on a computer-readable storage medium (page 8, lines 3-8). The computer program product includes instructions operable to cause a programmable processor to provide a representation of an XML document instance (page 4, lines 15-16; page 8, lines 14-24). The XML document instance contains two or more logical elements, where at least one logical element is a parent node and at least one logical element is a child node in a hierarchical tree structure describing the representation (page 4, lines 19-21). The computer program product includes instructions operable to cause the programmable processor to receive a query for logical elements satisfying an XPath expression (page 6, lines10-11; page 8, lines 14-24). The computer program product includes instructions operable to cause the programmable processor to search in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression (page 7, lines 1-3 and 18-26; page 8, lines 14-24). The computer program product includes instructions operable to cause the programmable processor to provide the logical elements satisfying the XPath expression (page 7, lines 24-26; page 8, lines 14-24).

(6) Grounds of Rejection to be Reviewed on Appeal

a. Objection to the Specification

The specification is objected to as failing to provide proper antecedent basis for the claimed subject under 37 CFR 1.75(d)(1) and MPEP § 608.01(o).

b. Rejection under 35 U.S.C. § 101

Claims 20-34 and 37-40 were rejected under 35 U.S.C. § 101 allegedly because "the computer readable medium recited in these claims also includes information carries, which are not tangible mediums."

c. Rejection under 35 U.S.C. § 102(b) over Chau

Claims 1-34 and 37-40 stand rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent Application Publication US2002/0156772A1 ("Chau").

(7) Argument

a. Objection to the Specification

The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter under 37 CFR 1.75(d)(1) and MPEP § 608.01(o). The examiner asserted that "the word 'tangible' in claim 20 is not present in the specification." *See* Office Action mailed March 8, 2007, page 3.

Claim 20 recites a "computer program product, tangibly encoded on a computer-readable storage medium." Antecedent basis for claim 20 can be found in the specification, for example, at lines 4-6 of page 8, which state that the "invention can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device…." Therefore, appellant respectfully submits that the specification provides proper antecedent basis for claim 20.

b. Rejection under 35 U.S.C. § 101

Claims 20-34 and 37-40 were rejected under 35 U.S.C. § 101 allegedly because "the computer readable medium recited in these claims also includes information carries [sic], which are not tangible mediums." *See* Office Action mailed March 8, 2007, page 2. The examiner suggested that the appellant "write 'computer readable storage medium' instead of 'computer readable medium'." *See* Office Action mailed March 8, 2007, page 2.

The appellant submitted an amendment under 37 CFR § 41.33 that was filed on July 30, 2007, to conform to the examiner's suggestion. In particular, the submitted amendment to claim

20 recited a "computer program product, tangibly encoded on a computer-readable storage medium." In an Advisory Action, the examiner stated that the submitted amendment would be entered and that the amendment overcomes the rejection under 35 U.S.C. §101. *See* Advisory Action mailed August 13, 2007, page 3.

c. Rejection under 35 U.S.C. §102(b) over Chau

Claim 1

Claim 1 recites a method for searching in a hierarchical tree structure contained in an XML document conforming to a schema used for XML. The method involves receiving a query for logical elements satisfying an XPath expression and searching in the hierarchical tree structure for those logical elements. The method only searches nodes that potentially have child nodes satisfying the XPath expression, in contrast to searching the entire tree of nodes.

FIGS. 2-4 illustrate searching in a hierarchical tree structure only nodes that potentially have child nodes satisfying an XPath expression. FIG. 2 shows a schematic view of a hierarchical representation of an XML document instance that represents a purchase order (200) (page 4, lines 22-23). FIG. 3 shows a graphical representation (300) of an XML schema for the instance of the purchase order shown in FIG. 2 (page 5, lines 3-4). FIG. 4 shows an example of searching for the XPath query, "find all city nodes of the XML document instance" (page 6, lines 14-15). In FIG. 4, the method searches only the dashed paths (405) (page 7, lines 27-28). The entire subtree under the "items" node (410) is avoided – according to the schema in FIG. 3, an "items" node does not potentially have child nodes that are "city" nodes (page 7, lines 29-30). Thus, the method results in faster searches of hierarchical trees contained in XML documents conforming to a schema (page 7, line 30 to page 8, line 2).

Chau does not anticipate claim 1, because Chau does not teach at least one element of claim 1: searching in a hierarchical tree structure only nodes that potentially have child nodes satisfying an XPath expression. Chau addresses four distinct methods related to inserting, retrieving, and searching XML documents in a relational database; these methods primarily show how to map XML documents onto relational database tables and back. It is understood that in all of these methods, where a hierarchical tree structure in an XML document conforming to a schema is traversed, the entire tree of nodes is traversed.

A. Chau's technique for creating metadata for searching XML documents stored in a table

The first method in Chau takes two steps: one, placing an XML document into a relational database; and two, searching the resulting database. Chau, paragraphs [0203]-[0322] (insertion and retrieval); paragraphs [0323]-[0358] (search).

The first step, insertion of an XML document (possibly conforming to a schema and containing a hierarchical tree structure) into a relational table, involves mapping the data from the XML document onto relational tables according to a Data Access Definition (DAD) document. Chau, paragraphs [0203]-[0206]. This step necessarily involves traversing all nodes of any trees contained in that XML document, rather than only nodes that potentially have child nodes satisfying an XPath Expression. After this step, the data from the XML document is no longer in a hierarchical tree structure; it is stored in a main table, and some attributes specified by the DAD are stored in side tables ("metadata"). Chau, paragraph [0211].

The second step, searching the data, involves receiving a query (possibly an XPath expression) and searching within the table for elements satisfying the query. Chau, paragraph [0211]. Chau teaches several ways to perform the search, all of which involve searching in a main table using indices created on side tables. Chau, paragraphs [0323]-[0358] ("Search from Join View", "Direct Query on Side Tables", "Query Using UDF", "Search on an Element or Attribute with Multiple Occurrences", "Structural-text Search"). These searches are different from the search described in claim 1 of the application. The search in claim 1 of the application involves directly searching a hierarchical tree structure in an XML document and searching only nodes that potentially have child nodes satisfying an XPath expression; in contrast, the searches taught in Chau involve searching in a table and using indices in side tables to locate the requested data.

The examiner relies upon the side table feature described by paragraphs [0075] and [0335]-[0336] of Chau to show "searching in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression." In particular, the examiner asserts that "the reference teaches that the side tables are made up of nodes in a hierarchical tree structure because the relational side tables contain elements or attributes of XML documents and there is mapping between an XML tree structure and relational tables." *See* Office Action mailed March 8, 2007, page 15.

The disclosure of side tables, however, fails to disclose the recited feature. The relied upon paragraphs do not show that side tables are a hierarchical tree structure. Further, the example side tables in Chau, paragraph [0268] are distinctly different from the hierarchical tree structure illustrated in FIG. 2 of the appellant's specification. As the examiner has not shown the "hierarchical tree structure" feature in any of the relied-upon paragraphs, a *prima facie* case of anticipation has not been made.

The examiner also relies on Chau's disclosure that the side tables can have "parent-children" relationships. Chau, paragraph [0335]. Even if Chau's remark about the side tables having "parent-children" relationships did mean that the side tables constituted nodes in a hierarchical tree structure (the appellant submits that Chau's remark does not mean this), the recited feature would not be met. If side tables constituted nodes, then meeting the recited feature would require searching only the side tables "that potentially have child [side tables] satisfying the XPath expression." The relied upon paragraphs say nothing about selectively choosing to search or not search a given side table. Thus, the side tables fail to meet the recited feature.

Therefore, it is understood that neither of the steps in this method in Chau disclose "searching in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression," as recited in claim 1.

B. Chau's technique for generating XML documents from a single SQL query

The second method in Chau involves generating XML documents from data in existing relational database tables using "SQL mapping." Chau, paragraph [0615]. This method does not involve searching an XML document containing a hierarchical data tree – it only involves retrieving data from a table and creating an XML document containing that data. Therefore, it is understood that this method in Chau does not disclose "searching in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression."

C. Chau's technique for generating XML documents from a relational database using XPath

The third method in Chau is similar to the second method but operates differently – it involves generating XML documents from data in existing relational database tables, but it uses the XPath data model. Chau, paragraph [0729]. Part of this method requires creating a Document Object Model (DOM), which is an XML document containing a hierarchical tree structure. Chau, paragraphs [0729]-[0730]. The method traverses the tree in the DOM to gather information, but it traverses the entire tree. It does not selectively examine only nodes that potentially contain child nodes satisfying an XPath expression. Therefore, it is understood that traversing all nodes in the DOM tree does not teach "searching in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression," as recited in claim 1.

D. Chau's technique for storing fragmented XML data into a relational database

The fourth method in Chau is another method for placing data from an XML document into relational database tables. Specifically, the method stores fragmented XML data into tables by decomposing (breaking down) data using a mapping scheme outlined in a DAD. Chau, paragraphs [0835]-[0838]. During this process, as in the third method, a DOM tree is created. Chau, paragraph [0836]. However, just as in the third method, whenever the DOM tree is traversed, the entire tree is traversed – every node. Therefore, it is understood that this method does not teach an element of claim 1: "searching in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression."

Thus, the cited portions of Chau cannot be read to teach or suggest "searching in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression," as recited in claim 1.

In view of the foregoing, the appellant respectfully requests reversal of the rejection of claim 1 under 35 U.S.C. §102(b).

Claims 2-19 depend from claim 1 and are therefore allowable for at least the same reasons as claim 1. Accordingly, appellant respectfully requests reversal of the rejection of claims 2-19 for at least the same reasons.

Claim 20

Claim 20 recites a computer program product directed towards searching for one or more logical elements in a hierarchical tree structure of an XML document conforming to a schema used for XML.

For at least the reasons stated above in reference to claim 1, Chau does not disclose a computer program product comprising instructions operable to cause a programmable processor to perform the steps recited in claim 20. In particular, it is understood that Chau does not disclose a computer program product comprising instructions operable to cause a programmable processor to "search in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression." As such, claim 20 is patentable over Chau and is in condition for allowance.

Accordingly, for the reasons noted above, appellant respectfully requests reversal of the rejection of claim 20 under 35 U.S.C. §102(b).

Claims 21-34 and 37-40 depend from claim 20 and are therefore allowable for at least the same reasons. Appellant respectfully requests reversal of the rejection under 35 U.S.C. §102(b) of claims 21-34 and 37-40 for at least the same reasons.

d. Conclusion and Relief

Based on the foregoing, the appellant requests that the Board overturn the examiner's rejection of all pending claims and hold that all of the claims of the present application are allowable.

In accordance with appellant's Notice of Appeal filed May 25, 2007, appellant submits this Appeal Brief. The Appeal Brief filing fee in the amount of $500 and the Petition for Extension of Time fee in the amount of $120 are being paid concurrently herewith on the Electronic Filing System (EFS) by way of Deposit Account authorization. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: August 22, 2007

Daniel J. Burns
Reg. No. 50,222

Customer No. 021876
Fish & Richardson P.C.
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

Appendix of Claims

1. A method for searching for one or more logical elements in a hierarchical tree structure of an extensible markup language (XML) document conforming to a schema used for XML, comprising:

providing a representation of an XML document instance containing two or more logical elements, wherein at least one logical element is a parent node and at least one logical element is a child node in a hierarchical tree structure describing the representation;

receiving a query for logical elements satisfying an XPath expression;

searching in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression; and

providing the logical elements satisfying the XPath expression.

2. The method of claim 1, including the further step of generating a collection of parent nodes that potentially have child nodes satisfying the XPath expression from a table relating a class of parent nodes and a class of child nodes, and wherein the table is used in the final searching step.

3. The method of claim 1, including the further step of generating a collection of parent nodes that potentially have child nodes satisfying the XPath expression from a table relating parent nodes and child nodes, and wherein the table is used in the searching step.

4. The method of claim 2, wherein the table comprises entries containing hash representations of a class of parent nodes and a class of child nodes.

5. The method of claim 3, wherein the table comprises entries containing hash representations of the parent nodes and child nodes.

6. The method of claim 2, wherein the table comprises a listing of permitted classes of child nodes for each class of parent node.

7. The method of claim 3, wherein the table comprises a listing of child nodes for each parent node.

8. The method of claim 2, wherein the table comprises a listing of permitted classes of parent nodes for each class of child node.

9. The method of claim 3, wherein table comprises a listing of permitted parent nodes for each child node.

10. The method of claim 1, further comprising:

receiving a rule set identifying allowable combinations between child nodes and parent nodes in a hierarchical document structure;

transforming the rule set into a table relating a class of parent nodes and a class of child nodes; and

using the table in the searching step.

11. The method of claim 1, further comprising:

receiving a rule set identifying allowable combinations between child nodes and parent nodes in a hierarchical document structure;

transforming the rule set into a table relating parent nodes and child nodes; and

using the table in the searching step.

12. The method of claim 10, wherein:

the rule set includes one of: an XML schema, a DTD, and a RelaxNg schema.

13. The method of claim 11, wherein:

the rule set includes one of: an XML schema, a DTD, and a RelaxNg schema.

14. The method of claim 2, wherein the table includes a listing of a not-permitted class of child nodes for each class of parent node.

15. The method of claim 3, wherein the table includes a listing of not-permitted child nodes for each parent node.

16. The method of claim 2, wherein the table includes a listing of a not-permitted class of parent nodes for each class of child node.

17. The method of claim 3, wherein the table includes a listing of a not-permitted parent nodes for each child node.

18. The method of claim 1, further comprising the additional steps of:

receiving a rule set identifying non-allowable combinations between child nodes and parent nodes in a hierarchical document structure; and

transforming the rule set into a table relating a class of parent nodes and a class of child nodes.

19. The method of claim 1, further comprising the additional steps of:

receiving a rule set identifying non-allowable combinations between child nodes and parent nodes in a hierarchical document structure; and

transforming the rule set into a table relating parent nodes and child nodes.

20. A computer program product, tangibly encoded on a computer-readable storage medium, for searching for one or more logical elements in a hierarchical tree structure of an extensible markup language (XML) document conforming to a schema used for XML, comprising instructions operable to cause a programmable processor to:

provide a representation of an XML document instance containing two or more logical elements, wherein at least one logical element is a parent node and at least one logical element is a child node in a hierarchical tree structure describing the representation;

receive a query for logical elements satisfying an XPath expression;

search in the hierarchical tree structure only nodes that potentially have child nodes satisfying the XPath expression; and

provide the logical elements satisfying the XPath expression.

21. The computer program product of claim 20, wherein the instructions cause a programmable processor to generate a collection of parent nodes that potentially have child nodes satisfying the XPath expression from a table relating a class of parent nodes and a class of child nodes, and wherein the instructions cause the table to be used in the search.

22. The computer program product of claim 20, wherein the instructions cause a programmable processor to generate a collection of parent nodes that potentially have child nodes satisfying the XPath expression from a table relating parent nodes and child nodes, and wherein the instructions cause the table to be used in the search.

23. The computer program product of claim 21, wherein the table comprises entries containing hash representations of the class of parent nodes and class of child nodes.

24. The computer program product of claim 22, wherein the table comprises entries containing hash representations of the parent nodes and child nodes.

25. The computer program product of claim 22, wherein the table comprises a listing of a permitted class of child nodes for each class of parent node.

26. The computer program product of claim 21, wherein the table comprises a listing of permitted child nodes for each parent node.

27. The computer program product of claim 21, wherein the table comprises a listing of a permitted class of parent nodes for each class of child node.

28. The computer program product of claim 22, wherein the table comprises a listing of permitted parent nodes for each child node.

29. The computer program product of claim 20, further comprising instructions to:

receive a rule set identifying allowable combinations between a class of child nodes and a class of parent nodes in a hierarchical document structure;

transform the rule set into a table relating the class of parent nodes and the class of child nodes; and use the table in the search.

30. The computer program product of claim 20, further comprising instructions to:

receive a rule set identifying allowable combinations between child nodes and parent nodes in a hierarchical document structure;

transform the rule set into a table relating parent nodes and child nodes; and use the table in the search.

31. The computer program product of claim 29, wherein:

the rule set includes one of: an XML schema, a DTD, and a RelaxNg schema.

32. The computer program product of claim 30, wherein:

the rule set includes one of: an XML schema, a DTD, and a RelaxNg schema.

33. The computer program product of claim 21, wherein the wherein the table includes a listing of a class of not-permitted child nodes for each class of parent node.

34. The computer program product of claim 22, wherein the wherein the table includes a listing of not-permitted child nodes for each parent node.

37. The computer program product of claim 21, wherein the wherein the table includes a listing of a class of not-permitted parent nodes for each class of child node.

38. The computer program product of claim 22, wherein the wherein the table includes a listing of not-permitted parent nodes for each child node.

39. The computer program product of claim 20, further comprising instructions to:

receive a rule set identifying non-allowable combinations between a class of child nodes and a class of parent nodes in a hierarchical document structure; and transform the rule set into a table relating the class of parent nodes and the class of child nodes.

40. The computer program product of claim 20, further comprising instructions to:

receive a rule set identifying non-allowable combinations between child nodes and parent nodes in a hierarchical document structure; and

transform the rule set into a table relating parent nodes and child nodes.

Evidence Appendix

None.

Related Proceedings Appendix

None.